RECEIVED
CENTRAL FAX CENTER
FEB 16 2007

IN THE SPECIFICATION:

Please amend the specification as follows:

Please amend page 2, lines 1-9 of the specification as follows:

UPC is a parallel extension to ANSI C, based on a distributed, shared-memory programing paradigm. It provides a common syntax and semantics for writing high-performance, explicitly parallel programs in C. An example of UPC is taught by T.A. El-Ghazawi, W. W. Carlson, and J. M. Draper, "UPC Language Specifications V1.1", issued March 24, 2003, as a joint work, by the University of California Berkerley and the National Energy Research Scientific Computing Center, Lawrence Berkely National Laboratory, Office of Science, US Department of Energy~~, and available in electronic form from http://upc.nersc.gov.~~

Please amend page 3, lines 2-5 of the specification as follows:

The gist of the invention is to generate C-level code strings for UPC constructs to be spliced into the source UPC program and then translated to intermediate form for optimization. Any UPC-unique components are discarded during ~~optamization~~ optimization, and the resultant intermediate form is then compiled as machine code.

Please amend page 3, lines 11-13 of the specification as follows:

The proxy ~~delcarations~~ declarations for a UPC statement include a conditional statement. For forall statements, the conditional statement has predicates leading to evaluation based upon an affinity test.

Please amend page 10, lines 29-32 of the specification as follows:

This algorithm ~~corresposnds~~ corresponds to the process described above with reference to **Fig. 3**.
Input: A UPC program which may have upc_forall statements in all generality.
Output: A translated version of the input program into an equivalent program that does not have upc_forall statements.

Please amend page 14, line 22 through page 15, line 2 of the specification as follows:

Besides reducing UPC-related code generation size, there is the advantage of being able to work largely at the human-readable, and well-known/standard C language level as opposed to a nonstandard/proprietary (i.e. a compiler vendor specific) low-level intermediate form, with its typical optimization-oriented features that are orthogonal to the ~~the~~ purpose of translating UPC code and only serve to make the task harder. The solution is able to reuse the compiler vendor's frontend code extensively, for translating the (generated) C strings to intermediate form and for taking care of optimization information. The automation in intermediate code generation that is able to be obtained thus reduces the complexity faced in digesting UPC annotations substantially. For example, all of the (extensive) initialization code for UPC data types is expressed completely

within C strings in the ~~disloosed~~ disclosed method and no intermediate code needs to be generated/manipulated directly for this purpose. The overall result is a minimally modified standard C compiler for that is also capable of translating UPC programs to executable object codes.